公聴会発表資料

Management of Object-Oriented Software Components in Distributed Environments

分散環境におけるオブジェクト指向
ソフトウェア部品管理に関する研究

知能システム学専攻
大月 美佳

平成11年2月22日
情報 S109教室

NEXT ▶

タイトル - 1

目次

◀ BACK NEXT ▶

目次 - 1

研究の背景と目的

アプリケーションの大規模化・複雑化

再利用による開発

ネットワークの普及

開発者の分散　部品の分散

分散環境で
再利用を支援する機構の必要性

1章 はじめに - 1

研究の背景と目的

再利用

定義

ソフトウェア開発に関わった全ての情報を
次のソフトウェア開発において利用すること

ソフトウェア部品

ソフトウェア開発の過程で生成・変更・参照
される全ての情報

利点

品質，生産性，効率，信頼性，相互運用性の向上

オブジェクト指向パラダイムにより部品化が促進

再利用 - 1

研究の背景と目的

オブジェクト指向パラダイム

オブジェクト指向言語

カプセル化、継承、多態性による部品化が可能

オブジェクト指向分析設計手法

プログラムの構造を明確に把握することが可能

いかに部品を作るかという指針は与えない

パターン言語

アプリケーションに繰り返し現れる構造
柔軟性・拡張性を導入するための知識
アーキテクチャ、デザインパターン、イディオム

オブジェクト指向パラダイム - 1

研究の背景と目的

オブジェクト指向ソフトウェア部品

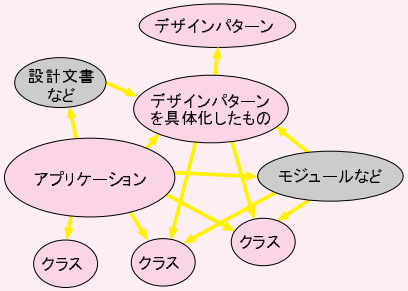
クラス

オブジェクト指向ソフトウェアの基本となる単位

デザインパターン

アプリケーションに繰り返して用いられる構造を一般化して記述したもの

再利用可能な構造を設計する基盤



デザインパターンを具体化したもの

クラスとデザインパターンを関連付けるのに必要

OO部品 - 1

研究の背景と目的

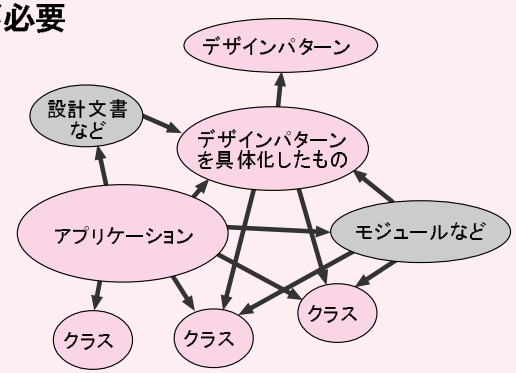
部品間の関係

クラス群を部品として再利用するためには，情報が必要

作られた仮定
構造情報
具体例

部品間の関連

背景知識の部品化



部品間の関係 - 1

研究の背景と目的

分散環境

分散開発環境の増加

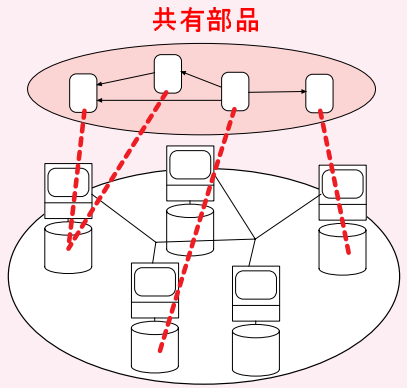
複数の企業が協同

開発者の分散

分散した部品間の関連

部品の種類・形式の多様化

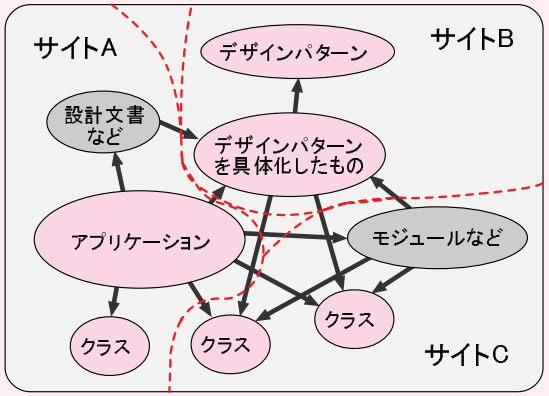
分散環境での再利用を促進するための枠組みが必要



分散環境 - 1

研究の背景と目的

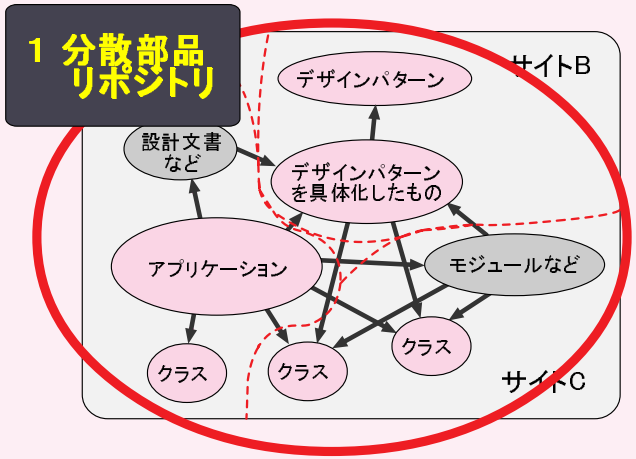
本研究の目的



本研究の目的 - 1

研究の背景と目的

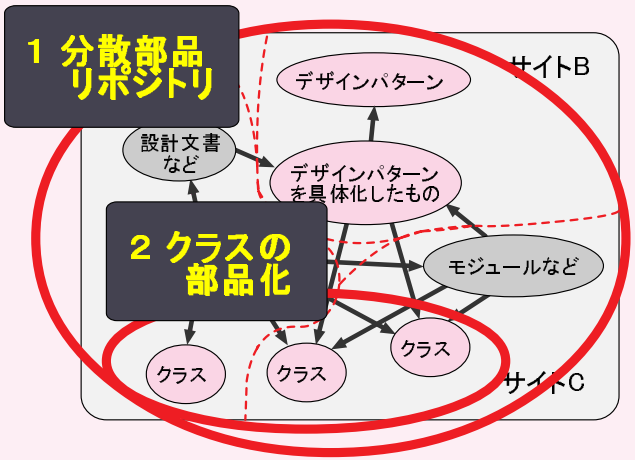
本研究の目的



本研究の目的 - 2

研究の背景と目的

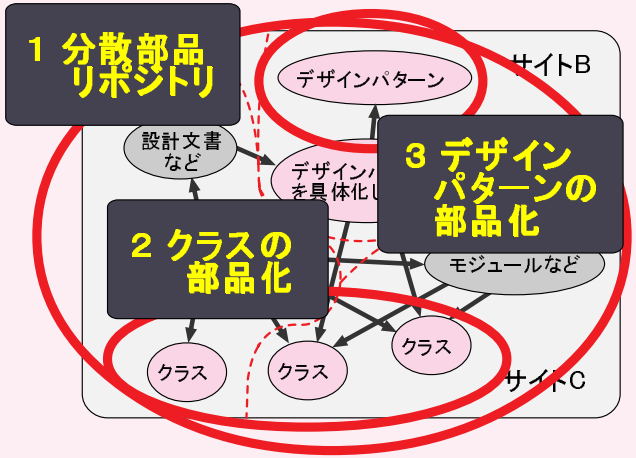
本研究の目的



本研究の目的 - 3

研究の背景と目的

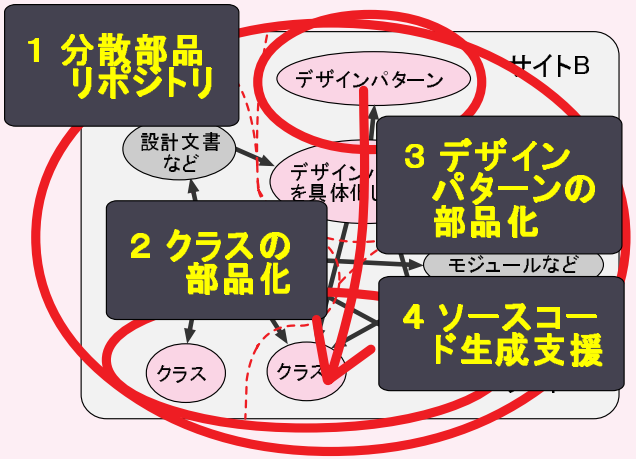
本研究の目的



本研究の目的 - 4

研究の背景と目的

本研究の目的



◀ BACK

NEXT ▶

本研究の目的 - 5

研究の背景と目的

本研究の目的 ―詳細―

クラス部品を対象にした分散部品リポジトリ

- 分散部品リポジトリ
- クラス部品抽出登録機構
  - プロトタイプ：C++を対象

デザインパターンの部品化

- 構造化文書による部品化の枠組み
- ソースコード生成支援機構
  - プロトタイプ：Javaを対象

◀ BACK

NEXT ▶

目的詳細 - 1

分散部品リポジトリ

1 要求
2 システム構成図
3 部品位置テーブルの一貫性維持
4 関連研究

◀ BACK

NEXT ▶

2章 分散部品リポジトリ - 1

分散部品リポジトリ

分散部品リポジトリへの要求

- 多様な部品に対して統一的な閲覧が可能
- 部品間の関連が辿れる
- 部品の分散に対応
  - → 分散ハイパーテキスト（WWW）の採用
- 名前の一貫性管理
- 分散透明性の実現
- 部品の位置変化に対する耐久性
  - → 部品位置管理サーバと問い合わせ用CGIプログラムの提供

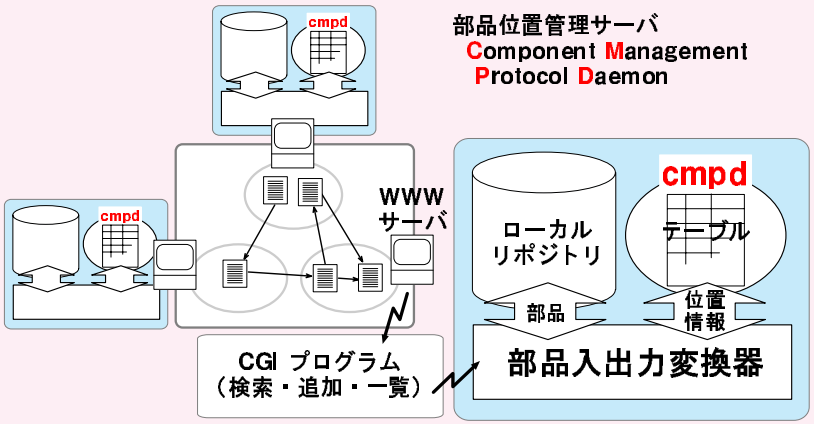
◀ BACK

NEXT ▶

要求 - 1

分散部品リポジトリ

システム構成図



システム構成図 - 1

分散部品リポジトリ

部品位置テーブルの一貫性維持

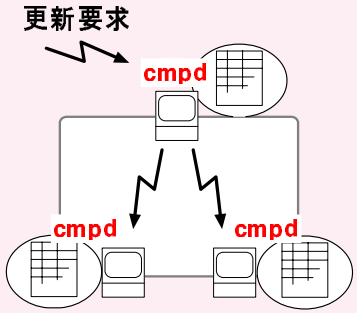
部品の識別子と位置の一貫性

部品位置管理サーバ同士の通信により維持

更新より参照の頻度が高い

↓

更新の度に全サーバに通知



一貫性維持 - 1

分散部品リポジトリ

関連研究

	Dynamic Design (米テクトロニクス社)	C++ のソースコードリポジトリ (立命館)	Chimera (カルフォルニア大)	本研究
表現形態	ハイパーテキスト	OODB	ハイパーテキスト	ハイパーテキスト
対象部品	従来の文書やモジュール	ソースコードのみ	任意のデータへの参照	クラス (パターン)
管理形態	集中管理型	集中管理型	集中管理型	分散対応
備考		解析機能に優れる	CASEの相互運用	

関連研究 - 1

クラス部品

3章 クラス部品 - 1

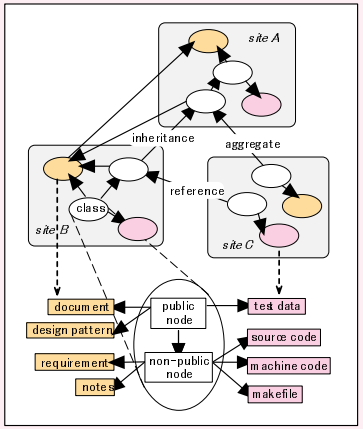
クラス部品

クラス部品

オブジェクト指向ソフトウェアの基本要素

他部品とのリンク
- 他のクラス
- 構成情報・ソースコード

2つのノードからなる
- 公開ノード
- 非公開ノード



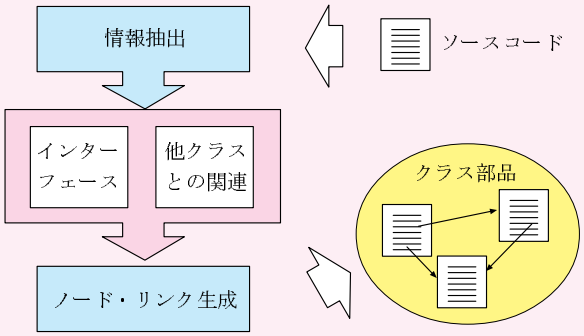
クラス部品とは - 1

クラス部品

ソースコードからの自動抽出機構

クラス部品の登録を簡便化

既存のプログラムからクラス部品を収集



自動抽出機構 - 1

クラス部品

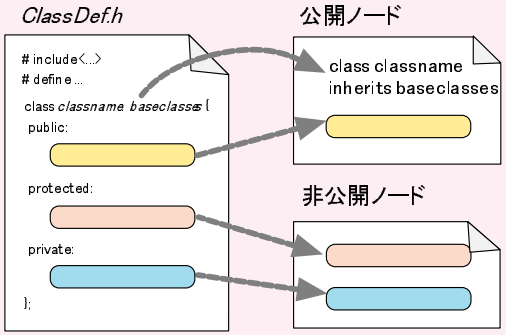
クラス情報の抽出

クラス情報
- クラス名
- 継承関係
- インターフェース
  - 関数と変数
  - 参照関係
- アクセスレベル

HTMLファイルとして生成

C++ のヘッダファイルから抽出

protected・private な情報は非公開ノードに



クラス情報抽出 - 1

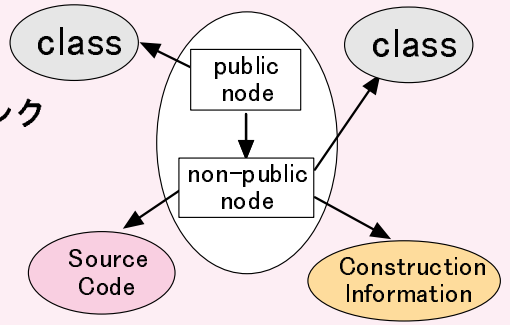
クラス部品

リンクの自動生成

公開ノードと非公開ノードとのリンク

他クラス部品へのリンク
- 継承関係・参照関係

他の部品へのリンク
- ソースコード
- 構築情報



部品名の動的解決

部品位置管理サーバに問い合わせて取得

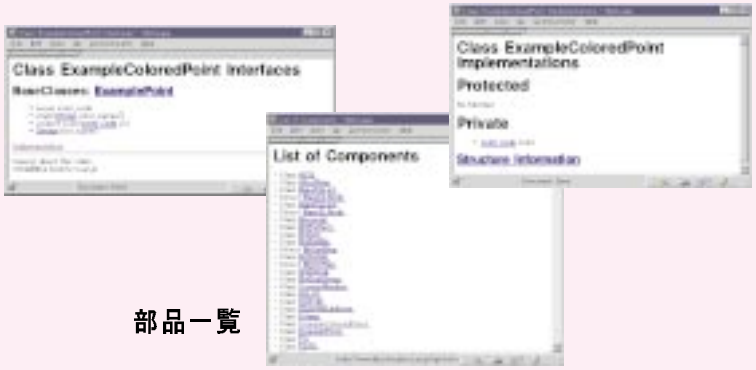
部品の位置変化に対して耐久性を提供

リンク - 1

生成例

公開ノード

非公開ノード



部品一覧

生成例 - 1

デザインパターン部品

4章 デザインパターン部品 - 1

デザインパターンとは

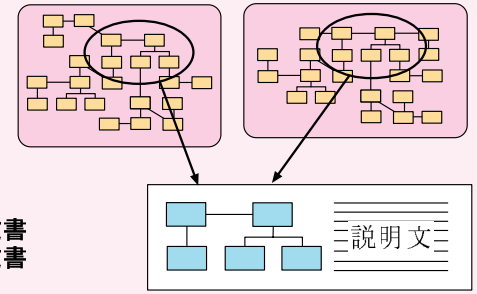
アプリケーションに共通する構造を抽出して
利用する理由や利用の方法，利用した結果などと
ともに文書化したもの

設計のノウハウ

意志疎通の基盤

従来の記述方式

論文や本などのオフライン文書
構造を持たないオンライン文書



平文（説明）＋クラス図（構造）＋疑似コード（振舞）

デザインパターンとは - 1

部品化への要求

デザインパターン部品化の促進
　整合性チェックが可能であること
　流通に適した形態であること

設計支援
　ソースコードの生成支援に利用できること

理解支援
　クラス群と対応づけが可能であること

構造化文書 → 文章・構造図・振舞いを一括して記述

部品化への要求 - 1

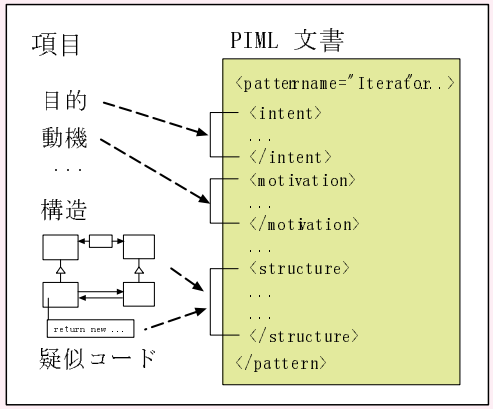
デザインパターン部品

デザインパターン記述言語（PIML）

PIML
= Pattern Information Markup Language

平文，図，疑似コードを統合的に扱うための記述言語

SGML (Standard General Markup Language) のインスタンス



◀ BACK　- 4-4 -　NEXT ▶

PIML - 1

デザインパターン部品

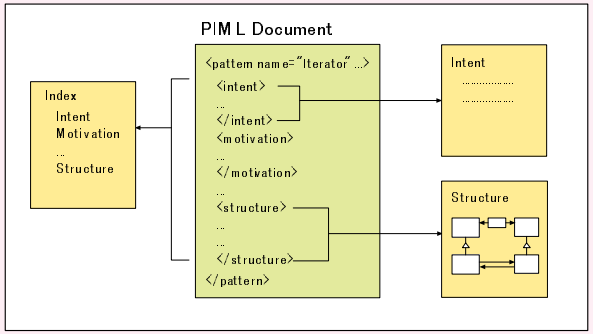
HTMLへの変換

閲覧のためのハイパーテキストノードの生成

構造情報からOMT図を生成

文書内の部品名→ 動的に解決

位置情報を文書から分離



◀ BACK　- 4-5 -　NEXT ▶

HTML変換 - 1

デザインパターン部品

生成画面例

目次

OMT図

「使用例」項目

◀ BACK　- 4-6 -　NEXT ▶

生成画面例 - 1

デザインパターン部品

関連研究

	Portland Pattern Repository	The DA VINCI Initiative (MIT)	IBM T. J. Watson 研究所	本研究
他部品へのリンク	静的	静的	静的	動的
構造図	なし	ビットマップ図	ビットマップ図＋マクロ	構造記述
振舞い	なし	なし	マクロ	疑似コード
統合しているか	文章のみ	統合していない	統合していない	統合

◀ BACK　- 4-7 -　NEXT ▶

デザインパターンの関連研究 - 1

ソースコード生成支援

◀ BACK - 5-1 - NEXT ▶

5章 ソースコード生成支援 - 1

目的と課題

生成支援の目的

設計支援 → 制約条件を保証し煩雑さを削減する

理解支援 → デザインパターンとクラス群の関連付け

生成の課題

パターン・クラス間の多対多関係

- 1アプリケーション 対 複数パターン
- 1アプリケーションクラス 対 複数パターンクラス
- 1パターンクラス 対 複数アプリケーションクラス

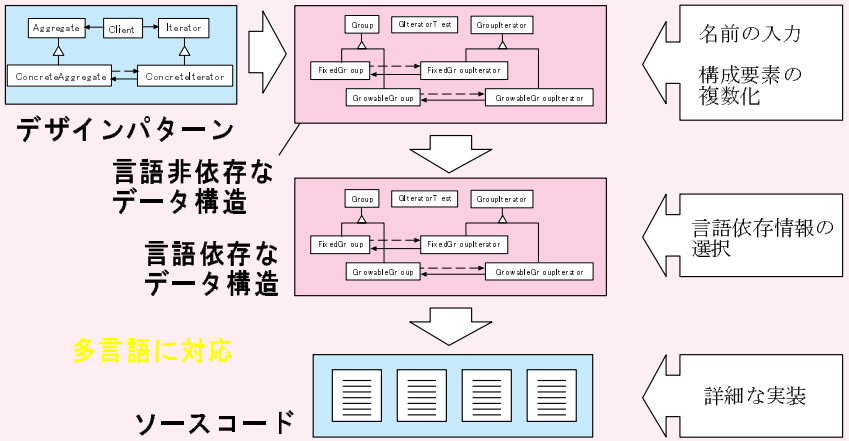
制約条件（一緒に複数化）の保証

多言語への対応 → 言語依存情報の分離

◀ BACK - 5-2 - NEXT ▶

目的と課題 - 1

生成の流れ



◀ BACK - 5-3 - NEXT ▶

生成の流れ - 1

実装

2種のデータ構造

言語非依存な情報と言語依存な情報

複数化の処理機構

PIMLに複数化の制約条件を記述

疑似コードから対象言語への変換機構

対話的インターフェース

プロトタイプ

対象言語：Java

言語依存部分を入れ換えることで，他言語にも対応可能

◀ BACK - 5-4 - NEXT ▶

実装 - 1

ソースコード生成支援

生成コード例

FixedGroupIterator

```
public class FixedGroupIterator implements Iterator {
  FixedGroup _group;
  int _cur_idx;

  public FixedGroupIterator (FixedGroup group) {
    _group = group;
  }

  public boolean isDone () {
    return _cur_idx >= _group.getMaxLength();
  }

  public void next () {
    _cur_idx++;
  }

  public void first () {
    _cur_idx = 0;
  }

  public Object getCurrentItem () {
    return _group.getAt(_cur_idx);
  }
}
```

パターンとして
必須のコード
を生成

Makefile

```
JAVAC = /usr/local/share/java/jjjalib/javac
JAVA = java
CLASSPATH = -classpath ./:/usr/local/share/java/jjjalib/classes.zip
JAVAOPT = $(CLASSPATH)
CLASSES = FixedGroupIterator.class FixedGroup.class \
          GrowableGroupIterator.class GrowableGroup.class \
          Group.class IteratorTest.class IteratorTest.class
%.class: %.java
        $(JAVAC) $(JAVAOPT) $<
all: myprog
myprog: $(CLASSES)
clean:
        rm -f *.class*
```

FixedGroup

```
public class FixedGroup implements Group {
  protected Object _elements[];
  protected int _max_length;
  protected int _cur_length;

  public FixedPersonGroup(int max_length) {
    persons = new Object[_max_length = max_length];
    _cur_length = 0;
  }

  public int getMaxLength() {
    return _max_length;
  }

  public void add(Object o) {
    if (_cur_length>=_max_length)
      errorHandle();
    else
      _elements[_cur_length++] = o;
  }

  public void removeLast() {
    if (_cur_length<=0)
      errorHandle();
    else
      _elements[--_cur_length] = null;
  }

  public Object getAt(int idx) {
    return elements[idx];
  }

  /**
   * auto generated
   */
  public Iterator createIterator () {
    return new FixedGroupIterator(this);
  }
}
```


生成コード例 - 1

ソースコード生成支援

関連研究

	Utrecht 大	IBM T. J. Watson 研究所	日本大学	本研究
テンプレート定義	オブジェクト	マクロ テンプレート	ROSE スケルトン	構造記述 ＋ 疑似コード
名前書き換え	手動	自動	手動？	自動
複数化への対応	事後チェック	なし	なし	自動
自動化率	低	中	中	中
多言語対応	なし	新規マクロ	不明	新規置換 モジュール

生成の関連研究 - 1

結論

6章 結論 - 1

結論

分散部品リポジトリ

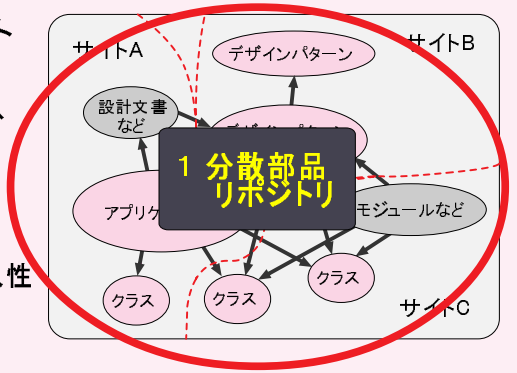
分散ハイパーテキスト

分散部品の関連
統一インターフェース

部品位置管理サーバ＋問い合わせ機能

分散透明性
位置変化に対する耐久性

部品位置テーブルの一貫性維持



分散部品リポジトリのまとめ - 1

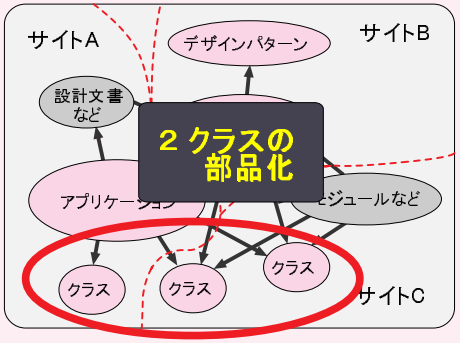
結論

クラス部品と自動抽出機構

基本部品として提供
- 他部品を統合する基盤

クラス抽出機構
- 既存資源からの収集

プロトタイプの実装
- C++ を対象
- 実用ライブラリで検証



クラス部品と自動抽出機構 - 1

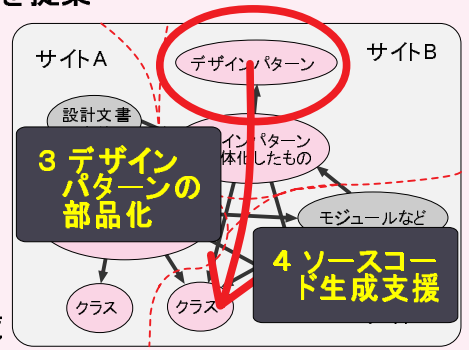
結論

デザインパターンの部品化

構造化文書による枠組みを提案

記述者
- 単一の文書として記述可
- 記述の整合性チェック可
- 電子文書流通に適

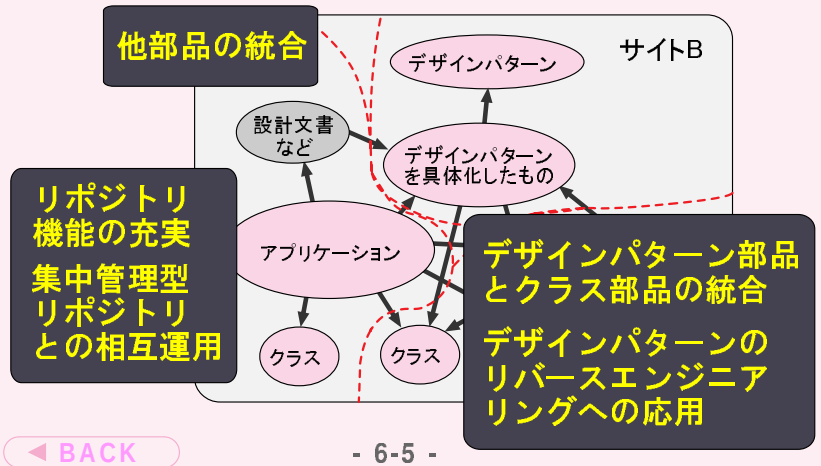
再利用者
- 部品として取得可
- 処理機構を提供
  - WWWブラウザでの閲覧
  - ソースコード生成支援



デザインパターンの部品化 - 1

結論

今後の課題



今後の課題 - 1